IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: )
)
ZE'EV DRORI )
)
Serial No. 08/118,167 )
)
Filed: September 8, 1993 )
)
For: ELECTRONICALLY PROGRAMMABLE )
REMOTE CONTROL ACCESS SYSTEM )
)

DECLARATION OF CARL ANGOTTI

Carl Angotti hereby states as follows:

1. I have reviewed the above referenced patent application (the "Patent Application") to determine if the material presented was adequate to enable a person skilled in the art of electronics, such as myself, at the time the application was first filed, which I understand to be September 8, 1987, to make and use the invention. Hereinafter, I will use the phrase "relevant time frame" to refer to the time frame prior to September 8, 1987.

Personal Background

2. I am a graduate Electrical Engineer, specializing in electronics engineering, with both a Bachelor and Masters Degree. I have more than 30 years of experience, and for about 18 years have been an independent systems and circuit design consultant. Appendix A sets forth a description of my background and work experience.

3. During the relevant time frame, I worked on numerous microprocessor-based electronic systems, and particularly since about 1984. Moreover, since September 8, 1987, I have also worked on many such systems. However, to my recollec-

tion, I have worked on only one vehicle security or alarm system which involved the use of a microprocessor, and that was in the 1987 time frame, prior to September 8, 1987, for a car and motorcycle alarm system. That design did not involve the decoding of signals as described in the Patent Application, but even prior to September 8, 1987, I was familiar generally with the encoding and decoding of information for transmission from one location to another for controlling electronic systems.

Time Spent on This Project

4. I spent approximately eight (8) hours reviewing the patent and the technology available in the relevant time frame, prior to September 8, 1987, for performing the functions described in the Patent Application. After this I spent about 20 hours defining how I would have accomplished these functions during the relevant time frame. First, I defined the context in which the encoding would be performed. Next, I assumed that a very straightforward, and easily generated software algorithm, with no special tricks or techniques, beyond the description in the Patent Application, would be devised to perform the functions described. From this, in about 8 hours, I produced flow diagrams, which could be given to a software programmer to generate the necessary firmware for use in a generic microcontroller. After this, I spent about twelve (12) hours to complete this report.

5. I have never met or had any discussion with the inventor, Ze'ev Drori, nor to my knowledge any employee of Clifford Electronics, Inc., regarding this invention or the Patent Application. I have had telephone discussions with an

attorney representing Clifford Electronics, Inc., in order to define the scope of this project.

Building the System

6. Generally, the Patent Application describes the hardware and software required to build a security system with the ability to handle single or multiple electronic keys. These keys consist of electronically coded signals that are generated by a encoder that is fed to a transmitter and then received by a receiver.

7. In most such systems in the relevant time frame, the value of the key code was set on the transmitter, and an identical key was programmed into a decoder at the receiver. When the two signals match, the encoder outputs that it has received a valid signal and this can be used to activate or deactivate the system, or perform some other function. In most systems available in the relevant time frame, the code was preprogrammed into the encoder and decoder. In addition, the length of the code was fixed to a specific number of bits. In the described invention of the Patent Application, the receiver can be set in a mode to learn the value of the encoded signal either for a specific number of bits or for a range of bits from N to M. It may then respond to these signals as valid when placed in its normal operating mode. This allows it to respond to many different encoder keys as valid. The operator need know nothing about programming the codes into either device.

8. The device may also be used to respond to more than one encoder key signal from a single transmitter. By altering the encoded signal in more than one learning mode for the receiver, the key can then perform more than one function.

For example in an automobile application, one key could disarm or arm an alarm, and the second could cause the trunk to open.

Description of Patent Application Circuit Diagrams

9. The circuit diagrams of Figures 3, 4 and 5 of the Patent Application illustrate an example of an encoder and transmitter and a receiver and decoder. The receiver, shown in Figure 4, removes the RF modulation of the carrier, and then sends the signal to a comparator with hysteresis. Assuming that there is a very high signal to noise ratio in the receiver, this comparator changes the input signal into a the final timed sequence that enters in the unmarked node of input to the Exclusive OR gate marked as 140 shown in Figure 5. This signal will closely resemble the encoded signal at the transmitter. The process of removing the information from this timed sequence is known as decoding. This decoding is performed in a microprocessor (MPU) that is also illustrated in Figure 5.

10. As was well known in the relevant time frame to those working with microprocessors, this signal to be decoded could enter a standard Port input on the MPU, or it could enter the specialized Interrupt pin. If it enters a port pin, it appears as a 1 or 0. If it enters an interrupt pin, the Exclusive OR gate 140 shown in Figure 5 of the Patent Application is significant. The other input to the Exclusive OR is programmed high or low via an output port by the software in the MPU. This will cause the digital signal to pass non--inverted or inverted to the Interrupt pin. This would allow it to respond with an interrupt of the software routine to either a positive or negative going transition, since these inputs normally only react to a transition on one edge. The

decoding process consists of responding to the timing of such transitions from either the Interrupt or standard Port input.

Practical Circuits

11. During the relevant time frame, there were straightforward and practical implementations of circuits that would perform the application described in the Patent Application. For such a circuit design, many generic microprocessor available in the relevant time frame could have been selected, since with a sufficiently high clock speed, any of them would have been effective. Potential devices suitable for the purpose of the microprocessor 142, Figure 5 of the Patent Application, include the following:

1. Mostek 6502
2. Motorola 6801, 6803
3. Intel 8086/88, 8048, 8051
4. Zilog Z80 and Z8

12. These microprocessor devices were readily available in the relevant time frame, among several others. Some of these such as the 6502, 6801/03, 8086/88 and Z80 would require peripheral support chips to perform the total functions required. This includes Read Only Memory (ROM), Random Access Memory (RAM) and Peripheral Support Port and Timer integrated circuits. With these support circuits, any of these devices could be selected. The other devices such as the 8048, 8051 or Z8 are single chip computers, later known as microcontrollers. They contain all or most of these functions internal to a single integrated circuit.

13. For the purposes of this review, the Intel 8051 would have been quite adequate. With a nominal oscillator frequency of 12 Mhz, the 8051 would have an internal cycle

time of 1 Mhz or 1 microsecond. There is nothing especially critical about selecting this circuit, except that it performs the required functions adequately, in a small package. Any of the other microprocessors could been used to perform the required function as well.

Encoders

14. During the relevant time frame, I was not aware of as many encoder ICs available to perform the encoder functions, but a moderately popular one is the Supertex ED9 device (See Appendix B for a complete specification). It has been available since 1982, and has been used, along with its derivatives as an encoder and in alarm applications. It contains both an encoder and a decoder on the same circuit. In addition to generating the encoded signal, it can perform the decoding function and provide an internal comparison of the received data signal to produce a level on an output if there is a match with a programmed input. This makes it an ideal device for a simple security alarm application. Of course, it can only detect a single security "key" signal. The invention of the Patent Application allows for the detection of several keys in a single receiver.

15. The decoder in the Supertex ED9 device can also be used to extract the data bits from the incoming data stream and store them in an external shift register, which can in turn, then present this data to a microprocessor port for further analysis. Instead of implementing the shift register in a hardware shift register, the data could also be presented to the microprocessor on a bit by bit basis to be stored in the internal MPU memory. Such implementations were well known in the relevant time frame.

16. In either of these cases, the data obtained could be stored in a data array of contiguous bytes. If a sequence of these bytes is stored in a special "learn" mode, it can later be compared byte by byte to another incoming data stream in the normal operating mode. If they match exactly, this would indicate a valid key that could be used to arm or disarm the security system.

17. This approach could have been used in to meet the requirements of the Patent Application. The microprocessor 142 in the receiver is set up to store a key for later comparison to future signals. This is the program mode described in the Patent Application. This approach would have met the requirements called out in the application for a system that utilizes a fixed number of data bits in the key because the ED9 device works with a fixed number of data bits. The flow chart for the software would be very simple, since the decoding is performed external to the microcontroller used in the system. This would have been an extremely straightforward storing and comparing software implementation.

18. It was possible during the relevant time frame to extend the approach to a slightly more complex system for which the number of the data bits in a key to vary over a given range such as M to N, where both M and N are predetermined constants. In such a case, the ED9 device is used as the encoder that is described in the patent, and a microprocessor can be used for the decoding function as is described in the application.

The ED9 Encoder and Microprocessor Based System

19. The ED9 encoder device output consists of a long transmission header of regular ones and zeros, followed by a

long sequence of zeros known as a start bit, followed again by a one and a string of data in the form of a series of ones and zeros, established by the programmed input. By a suitable connection, the device can emit a repeated series of the same encoded signal.

20. The Patent Application describes in Figure 6 a pulse width modulated signal that is used to modulate the transmitter carrier. Actually, as was well known in the relevant time frame, any consistent encoding method that allows for the later extraction of phase information for detection could be used to modulate the carrier. The ED9 device utilizes the well known Manchester encoding NRZ (Non Return to Zero) format, that allows for the adding of timing information, even if all zeros or ones are present in the data stream, while requiring less bandwidth than PWM modulation.

Encoder Timing

21. The exact signal encoding generated by the ED9 is illustrated in Figure C-1 attached hereto. A system that uses this chip in its transmitter would meet the requirements of an encoder as described in Figure 3 of the Patent Application. It produces a series of 12 ones and 11 zeros. This beginning signal could be referred to as a header or preamble. The preamble is followed by a long 9 count set of zeros followed by a single one bit. This represents the start bit of the data stream. After this, 15 data bits are sent, representing the "key" programmed on its input. Of course, a similar IC to the ED9 could be manufactured with a different number of data bits, say up to 32 bits.

Decoding with a Microprocessor

22. A circuit using a microprocessor such as the 8051 for decoding will now be described. This circuit could readily have been designed in the relevant time frame by a person that understood the use of microprocessors used as embedded controllers. For this example circuit, the following assumptions will be made:

1. The encoder baud rate is 1 kilobaud per second ±5%.
2. The encoder pulse to pulse jitter is less than ±5%.
3. The encoder signal repeats as long as the transmit button is held down with more than 100 milliseconds between transmissions during which time the data output is held at zero.
4. The encoder data bits will vary from 15 to 32.

23. This signal is then used to modulate a carrier as shown in Figure 3 of the Patent Application. By attaching the second button (54) to the LSB (Least Significant Bit) of the programmed input, two keys can be generated, one at the even key position, and one at the next highest odd key position. This is also shown in Figure 3 and allows for a dual function to be performed.

24. To perform the decoding function, two additional assumptions have been made. These are as follows:

1. The pulse distortion in the transmitter and receiver is less than +/- 5% of the total bit time.

2. The microprocessor timing is accurate to within +/- 1%, by utilizing a quartz crystal.

25. For purposes of the example circuit, the signal will be placed into a port input pin of the microprocessor. The signal at this input will consist of a sequence of ones and zeros that are timed at approximately 1 millisecond intervals.

The variations will be within less than about +/- 250 microseconds for the assumptions of this design. If the incoming signal is sampled at 250 microsecond intervals, most of the information contained in the signal can be captured. The next section describes how this information could be captured to determine if a key is valid or to have the processor store a new valid key.

Description of Firmware Flow Charts

26. In order to capture the incoming signals and use them for key storage or validity testing, software instructions must be added to the microprocessor, as was well known in the relevant time frame. An outline of a straightforward method for approaching this task, that would have been easily developed of by persons familiar with the general state of the art of such programs in the relevant time frame, is shown in attached Figures A-1 to A-5 and B-1. The task is to store the incoming serial data for later comparison to previously valid data. Figure A-1 shows the overall functions performed in the normal Receive and Compare function. The other diagrams illustrate what goes on in the general blocks.

27. In Figure A-1, there are two tasks that go on simultaneously. The time is divided between them, as was normally the practice in the relevant time frame. A greater percentage of the task is given to the receive function, since it is the most processing intensive. Data is received and processed. If a valid signal is received, a flag is set in the Receive and Store task. When this flag is set, the comparison task sets about to compare the received information to the valid keys previously stored. The signal is declared valid or invalid, and the balance of the code will act on this

information accordingly. This is accomplished by setting other flags that cause different actions in the software.

28. Figure A-2 illustrates the details of the Overall Receive and Store flow diagram. The remaining flow diagrams A-3 to A-5 show details of other portions of this flow chart. A-3 shows how the Header Valid Test is performed; Figure A-4 shows how the Start Valid Test is performed; and Figure A-5 shows how the Received key Comparison is performed.

29. Figure A-2 shows the receive diagram. First, the data is read each 250 microseconds; this is accomplished using the timer function in the MPU to interrupt the processor periodically (each 250 microseconds) to perform this task. The input data is read. This produces the data byte R(N), the current reading on the port pin. It is tested to the past reading. If it is the same, a counter, # Reads, is incremented for later use and R(N) is stored for use in comparing for the next read. If the two R(N) and R(N-1) are not equal, a signal transition has occurred. This causes a counter # Transitions to be incremented to also keep track of this count for later use. The counter and number of reads are tested in the Valid Header Test.

30. This is shown in detail in Figure A-3. First the process tests to see if the Header flag is valid, if it is, the balance of the process is bypassed. If it is not, then the remainder of is executed. A test is made of the value of # Read previously stored. If the number of reads is not less than 2 or greater than 6, the signal transition is invalid, and the loop is exited. This indicates that the incoming signal does not have approximately the correct baud rate and is therefore invalid. If it is between these values, the Header Count counter is incremented. This value is compared to 24, the number of valid transitions in the Header. If this

is true, The Header Flag is made valid, and the Header test will be bypassed on future inputs.

31. Returning to Figure A-2, if the Header is Invalid, the # Reads counter is set to zero and the loop waits for the next timing interrupt. If the Header is valid, the loop checks to see if the start is valid. This is detailed in Figure A-4. In this flow diagram, a test is made to see if a valid steam of zeros greater than 30 has occurred. If it has already occurred, this process is bypassed. If it has not, then the input is tested to assure that it is zero and that more than 30 of them have occurred. If so the Start Valid Flag is set for the next read input to bypass the testing. If the Value of R(N) = 0, then the process of reading continues until a "1" occurs. If the # Reads is not greater than 30, the Header Flag is reset, and the process must begin again from the validation of header. This indicates that if a Valid Header is not followed by a valid Start, the signal itself is not valid, and the receive must wait for another valid Header.

32. Returning again to Figure A-1, the testing now moves on to ready the serial data stream for storage. First a check is made to determine if the # Reads at this point is greater than 40 and the # Transitions is less than 88. If it is, the value of the data input R(N) and the # Reads are stored in two sequential bytes. This represents the value of the input, and the length of time that the input had this value. These are the information characteristics of the waveform at this point in time. The eventual value of this total array of data represents the "signature" of the incoming signal. If the value of the above test is true, this determines when to stop the reading of data into the data storage array. Either too many transitions have occurred, or a long series of zeros have been encountered indicating that the data transitions have

stopped. The number of transitions should not exceed the maximum number of transitions for a 32 bit data word, a valid Header and Start Sequence. If this is true, most likely the data key is invalid. If not, a valid Stop of the data has occurred and the data transmission is complete. At this time, the data in the Data Array #1 is copied into Array #2, along with the value of the counter # Transitions. Next, the Data Ready Flag is set to allow the comparison process to start. Then all of the Receiver flags are reset to zero, this includes Header Valid and Start Valid. In addition, the # Transitions counter is set to zero at this time. This will prepare the receiver to receive another data stream.

Received Key Comparison

33. This flow process is described in Figure A-5. Once the Data Ready Flag is set, the program can stop looping on this initial test. The Key is set to Key #1, to begin the comparison. Next the number of transitions in the stored in Array #2 is compared to that of Key #1, if this does not compare, the keys can not be identical, the Key # is incremented and the process is repeated until the # Transitions compares. If this occurs, then the elements of the array are compared one at a time to see if the values of R(N) are equal, and the # Reads stored compare to within +/- 1. This takes up the error introduced by variations in timing for the two signals. If there is no comparison, the Key # is again incremented. If the value of Key# Maximum is reached, then the Invalid Key Flag is set. After this, the Flags and Data are reset, including the Data Ready Flag, to wait for the next Data Ready. If the element by element comparison yields a

comparison at "Elements Compare +/-1", then the valid key is set and the Flags and Data are reset.

34. The balance of the software can use this Valid Key Flag to arm or disarm the alarm as was well known in the relevant time frame, or perform any other desired function.

Program Mode Flow Diagram

35. This flow process is described in Figure B-1. The program or "Learn" process is very similar to the receive process, except that a recovered key is stored in the next available Key# slot, Key#(Max) + 1. The operator causes the system to enter the "Learn" Mode. The Receive Data and Store process takes place as described earlier. When the Data Ready Flag is set, the "signature" obtained is stored in the next available key. Next, the operator presses the transmitter button a second time. Now the newly stored Key in Array #2 is compared to the Valid keys. A report is made to the operator if the newly learned "Key" is valid. If it is not, the operator receives an indication of this fact, the last Key Stored is dropped from the Valid Key List by setting Key#(Max) = Key#(Max) -1. The process then begins again. It continues until a valid Key "signature" is stored, or until the operator leaves the "Learn" Mode.

Conclusion

36. I believe that implementation of the security system described in the Patent Application could readily have been carried out, by one generally skilled in the microcomputer-controlled system art, using only the description set forth in the Patent Application, and with devices, general knowledge

and application skills readily available as of September 8, 1987. The design and construction of microcomputer-controlled systems such as this vehicle security system was well developed by September 8, 1987.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Dated: 10/17/94

Carl Angotti